

Cult3D

Project Terrain Watch

Goal

A first test of RT3D, using the Cult3D software to make an interactive 3D model of the watch, to be used in a commercial environment, showing its features and looks. JAVA implementation in Cult3D (setting the current time for the watch).

Approach

The Terrain watch was created using a 3D Modelling Software (3D Studio Max). First a picture of the real watch was scanned and used as a reference in 3D Studio Max. To get a realistic look we created some textures (image maps) to put on the 3D model. The image maps are again scanned images of the real watch, which were slightly altered in Photoshop to be used within our 3D Studio Max scene.

The exact same was done for the carton box and the case for the watch. Scanning the real objects and using them as textures for the 3D model.

After the modelling/texturing of the watch the Cult3D Exporter was used from within 3D Studio Max. The Cult3D Exporter saves the 3D model to a file format that can be read within Cult3D Designer (*.c3d). The Exporter window has some other features that were useful for our project.

As explained before the rule when it comes to RT3D on the Internet is to keep the file size of the 3D model to a minimum.

Because the watch had rather a lot of polygons its polygon-count had to be reduced. Using the Cult3D Exporter for optimizing the mesh (3D surface) we managed to reduce the amount of polygons (faces). Which increases the frame rate because the Cult3D engine (real-time renderer) has to calculate less. The frame rate of an animation is generally expressed in frames per second (fps). Specifically, this is the number of frames displayed for every second of real time movement. The fewer polygons in your mesh the faster the displaying.

Not only the mesh but also the texture maps could be compressed to get a small file size. Cult3D Exported then saved the optimized mesh for use in Cult3D Designer.

Now that we had our 3D model in a c3d-format, the next step was to import the 3D model into Cult3D Designer.

In Cult3D Designer we add all the interactivity. In this case the interactive features for the watch were:

- ~~///~~ Moving the carton away and back, revealing the watch's case.
- ~~///~~ Opening and closing the case, revealing the watch itself.
- ~~///~~ Manipulation of the watch:
 - adjusting the time.
 - Rotating the compass.
 - Rotating the watch.

Cult3D Designer is based upon event handling (for instance, a mouse click is an event). Every event is linked to one or more actions to be performed. In the case of the watch, clicking the carton box with the left mouse button would cause the carton to move away. Clicking is an event, moving is an action.

Interaction diagram

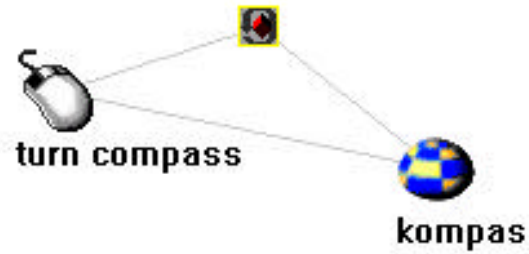
World start event
"world start"



world start

- ? sets the current time
- ? sets the viewport to camera
- ? zooms the camera
- ? triggers the start time event

Left Mouse Button event
"turn compass"



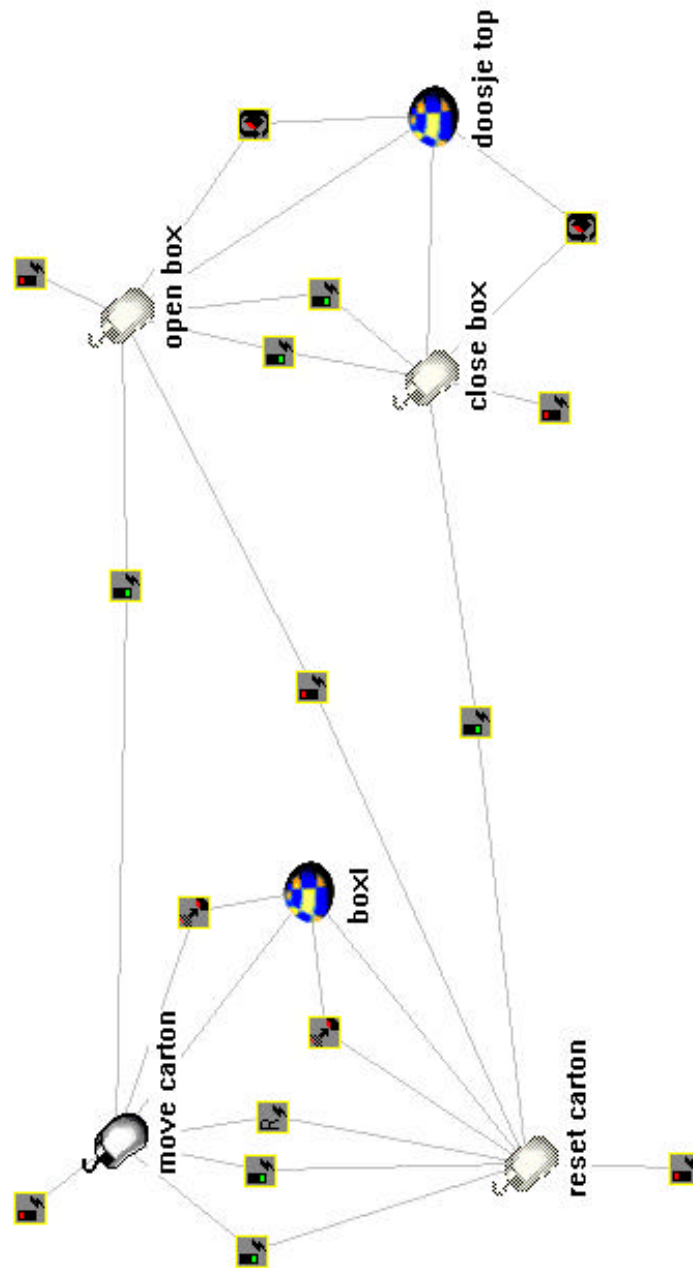
turn_compass
? compass rotation

Left Mouse Button event
"klik horloge"



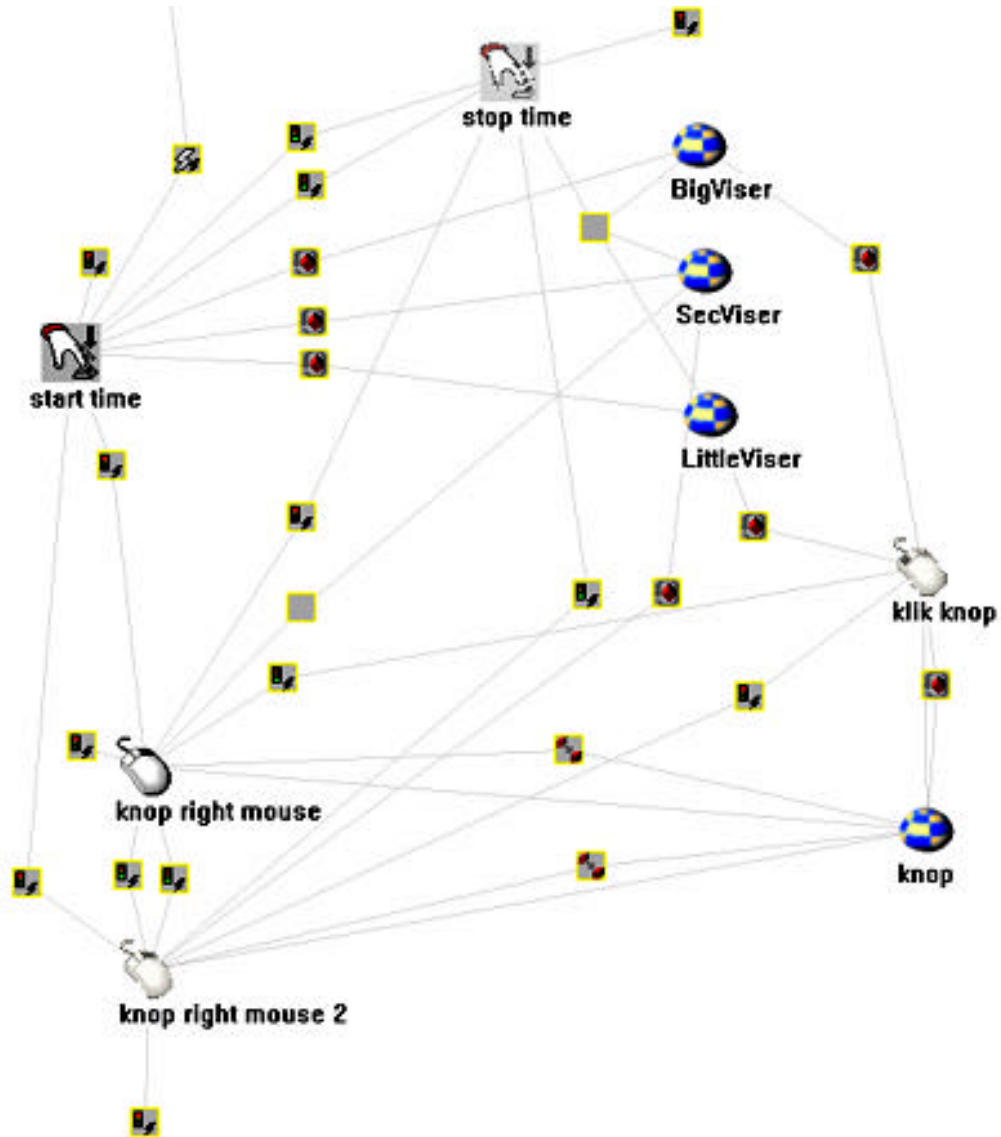
klik_horloge
? moves the watch to the front

Left Mouse Button events



- move carton
? moves the carton away
- reset carton
? moves the carton back
- open box
? opens the watch's box
- close box
? closes the watch's box

Keypress events & Mouse events



start time

? starts the time

stop time

? stops the time

knop right mouse

? pulls button to alter time

knop right mouse 2

? push button to resume time

klik knop

? alters the time with 15 minutes

JAVA

```
import com.Cult3D.Cult3DScript;
import com.Cult3D.world.CultObject;
import java.util.Date;

public class Clock extends Object implements Cult3DScript {

    private static final double TORADIANS;
    private CultObject sv; // Seconds
    private CultObject mv; // minutes
    private CultObject hv ; // hours

    public Clock()
    {
        TORADIANS = (Math.PI/180.0);
        sv = new CultObject("SecViser");
        mv = new CultObject("BigViser");
        hv = new CultObject("LittleViser");
    }

    // Set the current time
    public void setTime(String str)
    {
        // Get System.time
        Date d = new Date();
        int s = d.getSeconds();
        int m = d.getMinutes();
        int h = d.getHours();

        // position the visers
        sv.rotate (sv.Z, (float)(TORADIANS * -s*6));
        mv.rotate(mv.Z, (float)(TORADIANS * -m*6));
        hv.rotate(hv.Z, (float)(TORADIANS * -(h*30 + m*6)));
    }

    public void Cult3DDestroy()
    {
    }
}
```

Problems / Solutions

Problem:

How to automatically set the current time for the watch?

Solution:

Cult3D Designer has the possibility to import JAVA code. The integration of JAVA within Cult3D is based upon JAVA functions. JAVA functions that send/receive a string can be used in Cult3D Designer to link to events.

With Cult3D's own classes we can add more functionality to your project. This seemed to be very convenient to realize our goal.

With simple JAVA code we retrieved the current time and stored it in 3 variables (one for the hour, one for the minutes and one for the seconds).

Then a function from the Cult3D class "CultObject" called "rotate" was used to position the visers.

The class "CultObject" represents a 3D object in the Cult3D Designer.

To get the JAVA code working there had to be made some slight adjustments to the 3D model in 3D Studio Max.

All the visers should be pointed at 12 'o clock.

Why, well the JAVA code was based upon a formula that rotates the visers.

e.g.

```
// Get System.time
```

```
Date d = new Date();
```

```
int s = d.getSeconds();
```

```
int m = d.getMinutes();
```

```
int h = d.getHours();
```

```
// Use radians instead of degrees (Cult3D classes don't work with degrees)
```

```
private static final double TORADIANS = (Math.PI/180.0);
```

```
// rotate formula
```

```
sv.rotate (sv.Z, (float)(TORADIANS * -s*6));
```

```
mv.rotate(mv.Z, (float)(TORADIANS * -m*6));
```

```
hv.rotate(hv.Z, (float)(TORADIANS * -(h*30 + m*6)));
```

sv, mv and hv are cultobjects representing the 3D models of the visers in Cult3D Designer.

sv.Z means we are working around the Z.axis.

The final JAVA code was then imported into Cult3D Designer.

To get it working we linked our function "SetTime" with the "World Start" event.

"World Start" is an event that is activated automatically at start-up of the Cult3D model in the browser.

After adding all interactivity it was time to export it to a cultobject (.co) ready for the Internet. Therefore the exporter in Cult3D Designer was used.

Our optimized mesh could now be compressed even more using a special compression algorithm (Mesh Level 2). A high compression reduces the file size considerably, but may cause loss of quality in your 3D model.

At the final stadium the ".co file" was embedded in a HTML file to be viewed inside a browser.

Conclusion

The file size and image quality of a Cult3D object are well balanced. You can easily control the amount of compression/quality for your 3D object to be exported to Cult3D. With the Cult3D Designer it is very easy to add interactivity to your 3D models and to import JAVA code.